On the Complexity of Model-Checking Branching and Alternating-Time Temporal Logics in One-Counter Systems

Vester, Steen

Published in:
Proceedings of the 13th International Symposium on Automated Technology for Verification and Analysis, ATVA 2015

Link to article, DOI:
10.1007/978-3-319-24953-7_27

Publication date:
2015

Document Version
Peer reviewed version

On the Complexity of Model-checking Branching and Alternating-time Temporal Logics in One-counter systems

Steen Vester

Technical University of Denmark, Kgs. Lyngby, Denmark
stve@dtu.dk

Abstract. We study the complexity of the model-checking problem for the branching-time logic CTL* and the alternating-time temporal logics ATL/ATL* in one-counter processes and one-counter games respectively. The complexity is determined for all three logics when integer weights are input in unary (non-succinct) and binary (succinct) as well as when the input formula is fixed and is a parameter. Further, we show that deciding the winner in one-counter games with LTL objectives is 2ExpSpace-complete for both succinct and non-succinct games. We show that all the problems considered stay in the same complexity classes when we add quantitative constraints that can compare the current value of the counter with a constant.

1 Introduction

The branching-time temporal logics CTL* [11] and CTL [8] and the linear-time temporal logic LTL [19] are some of the most widely applied logics for automated verification. In particular, model-checking of these logics has been a very succesful approach [10]. The alternating-time temporal logics ATL and ATL* [1] extend these logics making it possible to reason about settings where several entities interact. The model-checking problem for alternating-time temporal logics subsumes the realizability problem for LTL [20, 21] which is the problem of deciding whether there exists a program satisfying a given LTL specification no matter how the environment behaves. This is closely related to the synthesis problem which consists of generating a program meeting such a specification. The model-checking problems have been studied quite extensively for finite-state systems, but for infinite-state systems there are still many interesting open problems.

In this work we focus on the model-checking problem for some of the simplest infinite-state systems one can construct, namely finite-state systems combined with an unbounded counter that can hold a non-negative integer value. The complexity of model-checking such systems has been determined for LTL [9, 12] and CTL [12, 13] but not yet for CTL*, ATL and ATL* which is the main purpose of this paper. Another focus of this paper is to consider generalizations of the logics capable of expressing combined qualitative and quantitative properties of

systems. This is done by extending to subsets of the quantitative alternating-time temporal logics QATL and QATL* [7] making it possible to compare the counter value with constants. This extension lets us express many interesting properties of systems in a simple way. As an example, consider deciding the winner in an energy game [2] modelling systems in which a controller needs to keep an energy level positive. This can be done by model-checking the QATL formula $\langle\langle \texttt{Ctrl} \rangle\rangle \mathbf{G}(r > 0)$ where r is used to denote the current value of the counter.

Let us give another example of a QATL specification. Consider the game in Figure 1 modelling the interaction between the controller of a vending machine and an environment. The environment controls the rectangular states and the controller controls the circular state. Initially, the environment can insert a coin or request coffee. Upon either input the controller can decrease or increase the balance, dispense coffee or release control to the environment again. Some

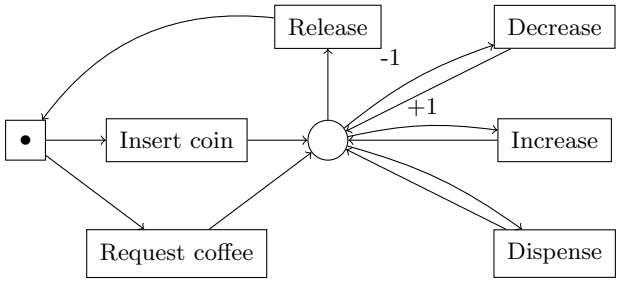

Fig. 1. Model of interaction between a vending machine controller and an environment.

examples of specifications in QATL* using this model are

- $\langle\langle \{\texttt{Ctrl}\} \rangle\rangle \mathbf{G}(\text{Request} \wedge (r < 3) \rightarrow \mathbf{XX}\text{Release})$: The controller can make sure that control is released immediately whenever coffee is requested and the balance is less than 3.
- $\langle\langle \{\texttt{Ctrl}\} \rangle\rangle \mathbf{G}(\text{Request} \wedge (r \geq 3) \rightarrow \mathbf{F}\text{Dispense})$: The controller can make sure that whenever coffee is requested and the balance is at least 3 then eventually a cup of coffee is dispensed.

It is indeed quite natural to model systems with a resource (e.g. battery level, time, money) using a counter where production and consumption correspond to increasing and decreasing the counter respectively. Extending to several counters would be meaningful, but as reachability games are already undecidable for games with two counters [6] the restriction to a single counter is very important.

1.1 Contribution

The contribution of this paper is to present algorithms and complexity results for model-checking of CTL*, ATL and ATL* in one-counter systems. The complexity

is investigated both in terms of whether only edge weights in $\{-1, 0, +1\}$ can be used (non-succinct systems) or if we allow any integer weights encoded in binary (succinct systems). We also distinguish between data complexity and combined complexity. In data complexity, the formula is assumed to be fixed whereas in combined complexity both the formula and the game are parameters.

We characterize the complexity of all the model-checking problems that arise from these distinctions. For CTL* the results on data complexity follow directly from results in [12, 22, 13] even though this is not mentioned explicitly. We also show that the logics considered can be extended with quantitative constraints without a jump in complexity and that deciding the winner in a one-counter game with LTL objectives is 2ExpSpace-complete as for ATL* model-checking in one-counter games in all cases considered. This closes a gap between 2ExpTime-completeness in finite-state games with LTL objectives and 3ExpTime-completeness in pushdown games with LTL objectives [17]. The results are presented below together with related results from the literature.

Table 1. Complexity of model-checking. All results are completeness results.

	Non-succinct		Succinct	
	Data	Combined	Data	Combined
CTL	PSpace [22, 13]	PSpace [22, 13]	ExpSpace [12]	ExpSpace [12]
CTL*	PSpace [22, 13]	ExpSpace	ExpSpace [12]	ExpSpace
μ-calculus	PSpace [22, 15, 14]	PSpace [22, 15, 14]	ExpSpace [12]	ExpSpace [12]
ATL	PSpace	PSpace	ExpSpace	ExpSpace
ATL*	PSpace	2ExpSpace	ExpSpace	2ExpSpace

Table 2. Complexity of deciding the winner in one-counter games with LTL objectives. All results are completeness results.

Non-succinct		Succinct	
Data	Combined	Data	Combined
PSpace	2ExpSpace	ExpSpace	2ExpSpace

1.2 Outline

In Section 2 we introduce the setting of the paper. In Section 3 model-checking algorithms based on model-checking games are presented. In Section 4 we provide lower bounds matching the complexity of the algorithms from Section 4. Section 5 provides concluding remarks.

2 Preliminaries

In this section we introduce the models, logics and problems considered.

2.1 One-counter games

A *one-counter game* (OCG) is a particular kind of finitely representable infinite-state game. Such a game is represented by a finite game graph where each transition is labelled with an integer from the set $\{-1, 0, 1\}$ as well as a counter that can hold any non-negative value. When a transition labelled v is taken and the current counter value is c, the counter value changes to $c + v$. We require that transitions are only applicable when $c + v \geq 0$.

Definition 1. *A one-counter game is a tuple* $\mathcal{G} = (S, \Pi, (S_j)_{j\in\Pi}, R, \mathrm{AP}, L)$ *where* S *is a finite set of states,* Π *is a finite set of players,* $\bigcup_{j\in\Pi} S_j$ *partitions* S*,* $R \subseteq S \times \{-1, 0, 1\} \times S$ *is a transition relation,* AP *is a finite set of propositions and* $L : S \to 2^{\mathrm{AP}}$ *is a labelling function.*

An OCG is played by placing a token in an initial state s_0 and then moving the token between states for an infinite number of rounds. The transitions must respect the transition relation and the intuition is that player j controls the successor state when the token is placed on a state in S_j. At any point of the game, the current counter value is given by the sum of the initial value $v_0 \in \mathbb{N}$ and all edge weights encountered so far.

More formally, an element $c \in S \times \mathbb{N}$ is called a *configuration* of the game. For a sequence $\rho = c_0c_1...$ of configurations we define $\rho_i = c_i$, $\rho_{\leq i} = c_0...c_i$ and $\rho_{\geq i} = c_ic_{i+1}...$. A *play* is a maximal sequence $\rho = (s_0, v_0)(s_1, v_1)...$ of configurations such that for all $i \geq 0$ we have $(s_i, v_{i+1} - v_i, s_{i+1}) \in R$ and $v_i \geq 0$. A *history* is a proper prefix of a play. The set of plays and histories in an OCG $\mathcal{G}$ are denoted by $\mathrm{Play}_\mathcal{G}$ and $\mathrm{Hist}_\mathcal{G}$ respectively.

A *strategy* for player $j \in \Pi$ in $\mathcal{G}$ is a partial function $\sigma : \mathrm{Hist}_\mathcal{G} \to S \times \mathbb{N}$ defined for all histories $h = (s_0, v_0)...(s_\ell, v_\ell)$ s.t. $s_\ell \in S_j$ requiring that if $\sigma(h) = (s, v)$ then $(s_\ell, v - v_\ell, s) \in R$. A play (resp. history) $\rho = c_0c_1...$ (resp. $\rho = c_0...c_\ell$) is compatible with a strategy σ_j for player $j \in \Pi$ if $\sigma_j(\rho_{\leq i}) = \rho_{i+1}$ for all $i \geq 0$ (resp. $0 \leq i < \ell$) such that $\rho_i \in S_j \times \mathbb{N}$. For a coalition $A \subseteq \Pi$ of players a collective strategy $\sigma = (\sigma_j)_{j\in A}$ is a tuple of strategies, one for each player in A. We denote by $\mathrm{Strat}^A_\mathcal{G}$ the set of collective strategies of coalition A. For an initial configuration c_0 and collective strategy $\sigma = (\sigma_j)_{j\in A}$ we denote by $\mathrm{Play}_\mathcal{G}(c_0, \sigma)$ the plays with initial configuration c_0 compatible with σ_j for every $j \in A$.

We extend one-counter games so arbitrary integer weights are allowed and such that transitions are still disabled if they would make the counter go negative. Such games are called *succinct one-counter games* (SOCGs). The special cases of OCGs and SOCGs where $|\Pi| = 1$ are called *one-counter processes* (OCPs) and *succinct one-counter processes* (SOCPs) respectively. In these cases we omit Π and $(S_j)_{j\in\Pi}$ from the definition.

By a *one-counter parity game* (OCPG) we mean a one-counter game with two players and a parity winning condition. It was shown in [22] that the winner can be determined in an OCPG in PSpace by reducing to the emptiness problem for alternating two-way parity automata [24].

Proposition 1. *Determining the winner in OCPGs is in* PSpace.

2.2 Temporal logics

The logics considered are fragments of alternating-time temporal logic ATL* [1] interpreted in one-counter games. The formulas of ATL* are defined by

$$\Phi ::= p \mid \neg\Phi_1 \mid \Phi_1 \vee \Phi_2 \mid \mathbf{X}\Phi_1 \mid \Phi_1 \mathbf{U} \Phi_2 \mid \langle\langle A \rangle\rangle \Phi_1$$

where $p \in \mathrm{AP}$, $A \subseteq \Pi$ and Φ_1, Φ_2 are ATL* formulas. We define the syntactic fragment ATL of ATL* by the grammar

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \langle\langle A \rangle\rangle \mathbf{X}\varphi_1 \mid \langle\langle A \rangle\rangle \mathbf{G}\varphi_1 \mid \langle\langle A \rangle\rangle \varphi_1 \mathbf{U} \varphi_2$$

where $p \in \mathrm{AP}$, $A \subseteq \Pi$ and φ_1, φ_2 are ATL formulas. The branching-time logics CTL [8] and CTL* [11] are the fragments of ATL and ATL* respectively such that $|\Pi| = 1$. In this case the universal and existential path quantifiers are defined by $\mathbf{A} = \langle\langle \emptyset \rangle\rangle$ and $\mathbf{E} = \langle\langle \{\mathrm{I}\} \rangle\rangle$ where $\Pi = \{\mathrm{I}\}$. LTL [19] is the fragment of CTL* with no path quantifiers. We interpret formulas of ATL* in OCGs. In ATL* we have state formulas and path formulas which are interpreted in configurations and plays of the OCG respectively. For definitions of state and path formulas, see [1]. The semantics of a formula is defined with respect to a given OCG $\mathcal{M} = (S, \Pi, (S_j)_{j \in \Pi}, R, \mathrm{AP}, L)$. For all states $s \in S$, plays $\rho \in \mathrm{Play}_{\mathcal{M}}$, $p \in \mathrm{AP}$, $i \in \mathbb{N}$, $A \subseteq \Pi$, ATL* state formulas Φ_1, Φ_2 and ATL* path formulas Ψ_1, Ψ_2 define

$$\begin{array}{lll}
\mathcal{M}, s, i \models p & \text{iff } p \in L(s) \\
\mathcal{M}, s, i \models \neg\Phi_1 & \text{iff } \mathcal{M}, s, i \not\models \Phi_1 \\
\mathcal{M}, s, i \models \Phi_1 \vee \Phi_2 & \text{iff } \mathcal{M}, s, i \models \Phi_1 \text{ or } \mathcal{M}, s, i \models \Phi_2 \\
\mathcal{M}, s, i \models \langle\langle A \rangle\rangle \Psi_1 & \text{iff } \exists \sigma \in \mathrm{Strat}^A_{\mathcal{M}}. \forall \pi \in \mathrm{Play}_{\mathcal{M}}((s,i), \sigma). \mathcal{M}, \pi \models \Psi_1 \\
\mathcal{M}, \rho \models \Phi_1 & \text{iff } \mathcal{M}, \rho_0 \models \Phi_1 \\
\mathcal{M}, \rho \models \neg\Psi_1 & \text{iff } \mathcal{M}, \rho \not\models \Psi_1 \\
\mathcal{M}, \rho \models \Psi_1 \vee \Psi_2 & \text{iff } \mathcal{M}, \rho \models \Psi_1 \text{ or } \mathcal{M}, \rho \models \Psi_2 \\
\mathcal{M}, \rho \models \mathbf{X}\Psi_1 & \text{iff } \mathcal{M}, \rho_{\geq 1} \models \Psi_1 \\
\mathcal{M}, \rho \models \Psi_1 \mathbf{U} \Psi_2 & \text{iff } \exists k \geq 0. \mathcal{M}, \rho_{\geq k} \models \Psi_2 \text{ and } \forall 0 \leq j < k. \mathcal{M}, \rho_{\geq j} \models \Psi_1
\end{array}$$

The semantics is extended in the natural way to SOCGs. In LTL, CTL and CTL* the formulas are interpreted in OCPs and SOCPs as defined above.

As an extension, we consider fragments of the quantitative alternating-time temporal logics QATL and QATL* [7]. These logics extend ATL and ATL* with state formulas of the form $r \bowtie c$ where $c \in \mathbb{Z}$, $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ and $k \in \mathbb{N}$. This type of formula is called a counter constraint and is interpreted such that e.g. $r \leq 5$ is true if the current counter value is at most 5 whereas $r \equiv_4 3$ is true if the current counter value is equivalent to 3 modulo 4. Formally, the semantics is given by $\mathcal{M}, s, i \models r \bowtie c$ iff $i \bowtie c$. The extension of $\mathcal{L} \in \{\mathrm{LTL}, \mathrm{CTL}, \mathrm{CTL}^*\}$ with counter constraints is called Q$\mathcal{L}$ as for QATL*.

In this paper we focus on the model-checking problem. That is to decide, given an OCG/SOCG $\mathcal{M}$, a state s in $\mathcal{M}$, a natural number i and a state formula φ whether $\mathcal{M}, s, i \models \varphi$. When model-checking non-succinct models, the initial counter value, edge weights and integers in the formula are assumed to be input in unary. For succinct models they are input in binary.

3 Model-checking algorithms

When model-checking branching and alternating-time temporal logics in finite-state systems, the standard approach is to process the state subformulas from the innermost to the outermost, at each step labelling states where the subformula is true. This approach does not work directly in our setting since the state space is infinite. We therefore take a different route and develop a model-checking game in which we can avoid explicitly labelling the configurations in which a subformula is true. This approach gives us optimal complexity in all cases considered and also allows us to extend to quantitative constraints in a natural way. We first present the approach for ATL and afterwards adapt it to ATL* and CTL* by combining it with automata on infinite words. Finally it is shown how to handle counter constraints.

3.1 A model-checking game for ATL

We convert the model-checking problem asking whether $\mathcal{M}, s_0, i \models \varphi$ for an ATL formula φ and OCG $\mathcal{M} = (S, \Pi, (S_j)_{j\in\Pi}, R, \mathrm{AP}, L)$ to a model-checking game $\mathcal{G}_{\mathcal{M},s_0,i}(\varphi)$ between two players Verifier and Falsifier that are trying to respectively verify and falsify the formula. The construction is done so Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s_0,i}(\varphi)$ if and only if $\mathcal{M}, s_0, i \models \varphi$ and is done inductively on the structure of φ. For a given ATL formula, a given OCG $\mathcal{M}$ and a given state s in $\mathcal{M}$ we define a characteristic OCG $\mathcal{G}_{\mathcal{M},s}(\varphi)$. Note that the initial counter value is not present in the construction yet. When illustrating the games, the circle states are controlled by Verifier, square states are controlled by Falsifier and diamond states can be both depending on the game. States with color 1 are filled and states with color 0 are not. Verifier wins the game if the least color that appears infinitely often during the play is even, otherwise Falsifier wins the game. The edges are labelled with counter updates, but 0-labels are omitted. The construction is done as follows.

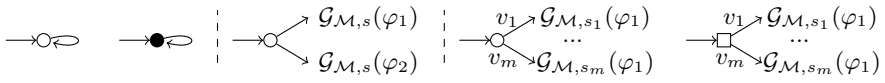

Fig. 2. $\mathcal{G}_{\mathcal{M},s}(p)$ to the left for $p \in L(s)$ and $p \notin L(s)$. $\mathcal{G}_{\mathcal{M},s}(\varphi_1 \vee \varphi_2)$ is in the middle. $\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A\rangle\!\rangle \mathbf{X}\varphi_1)$ to the right in the cases where $s \in \bigcup_{j\in A} S_j$ and where $s \notin \bigcup_{j\in A} S_j$.

- $\mathcal{G}_{\mathcal{M},s}(p)$: There are two cases which are illustrated in Figure 2 to the left.
- $\mathcal{G}_{\mathcal{M},s}(\varphi_1 \vee \varphi_2)$: The game is shown in Figure 2 in the middle.
- $\mathcal{G}_{\mathcal{M},s}(\neg\varphi_1)$: The game is constructed from $\mathcal{G}_{\mathcal{M},s}(\varphi_1)$ by interchanging circle states and square states and either adding or subtracting 1 to/from all colors.

- $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \mathbf{X}\varphi_1)$: Let $R(s) = \{(s, v, s') \in R\} = \{(s, v_1, s_1), ..., (s, v_m, s_m)\}$. There are two cases to consider. One when $s \in S_j$ for some $j \in A$ and one when $s \notin S_j$ for all $j \in A$. Both are illustrated in Figure 2 to the right.
- $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$: We let $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ have the same structure as $\mathcal{M}$, but with a few differences. Verifier controls all states that are in $\bigcup_{j\in A} S_j$ and Falsifier controls the rest. Further, for each transition $t = (s', v, s'') \in R$ we add a state s_t controlled by Falsifier between s' and s''. If the player controlling s' chooses transition t the play is taken to the state s_t from which Falsifier can either choose to continue to s'' or to $\mathcal{G}_{\mathcal{M},s''}(\varphi_1)$. Every state in $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ which is not part of $\mathcal{G}_{\mathcal{M},s''}(\varphi_1)$ has color 0. It is illustrated in Figure 3. The intuition is that Falsifier can challenge and claim that φ_1 is not true in the current configuration. If he does so, Verifier must be able show that it is in fact true in order to win. In addition, such a module is added before the initial state s so Falsifier can challenge the truth of φ_1 already in the initial state.
- $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \varphi_1 \mathbf{U} \varphi_2)$: The game is similar to the case of $\langle\langle A\rangle\rangle \mathbf{G}$. The differences are that every state is colored 1 and for each transition $t = (s', v, s'') \in R$ we add two states s_t and s'_t controlled by Verifier and Falsifier respectively with transitions to $\mathcal{G}_{\mathcal{M},s''}(\varphi_2)$ and $\mathcal{G}_{\mathcal{M},s''}(\varphi_1)$ respectively. It is illustrated in Figure 3. The intuition is similar, but here Verifier loses unless he can claim φ_2 is true at some point (and subsequently show this). In addition φ_1 cannot become false before this point, because then Falsifier can claim φ_1 is false and win. As for the previous case, such a module is added before the initial state as well.

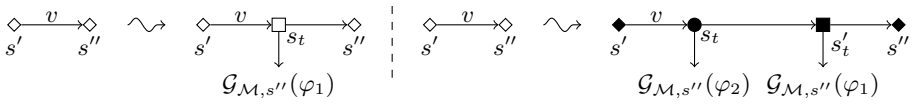

Fig. 3. $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ and $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle \varphi_1 \mathbf{U} \varphi_2)$ are obtained by updating each transition in $\mathcal{M}$ as shown to the left and right respectively.

Finally, we define the game $\mathcal{G}_{\mathcal{M},s,i}(\varphi)$ from $\mathcal{G}_{\mathcal{M},s}(\varphi)$ and $i \in \mathbb{N}$ by adding an initial module such that the counter is increased to i before entering $\mathcal{G}_{\mathcal{M},s}(\varphi)$. We can now show the following by induction on the structure of the ATL formula.

Proposition 2. *For every OCG $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and $\varphi \in$ ATL*

$$\mathcal{M}, s, i \models \varphi \text{ if and only if Verifier has a winning strategy in } \mathcal{G}_{\mathcal{M},s,i}(\varphi)$$

Proof. The proof is done by induction on the structure φ. Due to space limitations we only present the proof for the case where $\varphi = \langle\langle A\rangle\rangle \mathbf{G}\varphi_1$. To this end assume that the proposition is true for the subformula φ_1.

The intuition of the construction is that Verifier controls the players in A and Falsifier controls the players in $\Pi \setminus A$. At each configuration $(s', v) \in S \times \mathbb{N}$ of the game Falsifier can challenge the truth value of φ_1 by going to $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ in which Falsifier has a winning strategy if and only if φ_1 is indeed false in $\mathcal{M}, s', v$ by the induction hypothesis. If Falsifier challenges at the wrong time or never challenges then Verifier can make sure to win.

More precisely, suppose Verifier has a winning strategy σ in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ then every possible play when Verifier plays according to σ either never goes into one of the modules $\mathcal{G}_{\mathcal{M},s'}(\varphi_1)$ or the play goes into one of the modules at some point and never returns. Since σ is a winning strategy for Verifier, we have by the induction hypothesis that every pair $(s', v) \in S \times \mathbb{N}$ reachable when Verifier plays according to σ is such that $\mathcal{M}, s', v \models \varphi_1$, because otherwise σ would not be a winning strategy for Verifier. If coalition A follows the same strategy σ adapted to $\mathcal{M}$ then the same state, value pairs are reachable. Since for all these reachable pairs (s', v) we have $\mathcal{M}, s', v \models \varphi_1$ this strategy is a witness that $\mathcal{M}, s, i \models \langle\langle A\rangle\rangle \mathbf{G}\varphi_1$.

On the other hand, suppose that coalition A can ensure $\mathbf{G}\varphi_1$ from (s, i) using strategy σ. Then in every reachable configuration (s', v) we have $\mathcal{M}, s', v \models \varphi_1$. From this we can generate a winning strategy for Verifier in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ that plays in the same way until (if ever) Falsifier challenges and takes a transition to a module $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ for some (s', v). Since the same configurations can be reached before a challenge as when A plays according to σ, this means that Verifier can make sure to win in $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ by the induction hypothesis. Thus, if Falsifier challenges Verifier can make sure to win and if Falsifier never challenges Verifier also wins since all states reached have color 0. Thus, Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$. □

We can create a model-checking game for ATL in SOCGs as for OCGs and obtain a model-checking game which is a succinct OCPG. This can be transformed into an OCPG that is exponentially larger. It is done by replacing each transition with weight v with a path that has $|v|$ transitions and adding small gadgets so a player loses if he takes a transition with value $-w$ for $w \in \mathbb{N}$ when the current counter value $r < w$. The exponential blowup is due to weights being input in binary. By Proposition 1 this gives upper bounds for ATL model-checking. Matching lower bounds follow from PSPACE-hardness [13] and EXPSPACE-hardness [12] of data complexity of CTL in OCPs and SOCPs respectively.

Theorem 1. *The data complexity and combined complexity of model-checking* ATL *are* PSPACE*-complete for OCGs and* EXPSPACE*-complete for SOCGs.*

3.2 Adapting the construction to ATL*

As for ATL we rely on the approach of a model-checking game when model-checking ATL*. However, due to the extended possibilities of nesting we do not handle temporal operators directly as for ATL. Instead, we resort to translation of LTL formulas into deterministic parity automata (DPA) which are combined

with the model-checking games. This gives us model-checking games which are one-counter parity games as for ATL, but with a doubly-exponential blowup.

Let $\mathcal{M} = (S, \Pi, (S_j)_{j\in\Pi}, R, AP, L)$ be an OCG, $s_0 \in S$, $i \in \mathbb{N}$ and φ be an ATL* state formula. The algorithm to decide whether $\mathcal{M}, s_0, i \models \varphi$ follows along the same lines as our algorithm for ATL. That is, we construct a model-checking game $\mathcal{G}_{\mathcal{M},s_0,i}(\varphi)$ between Verifier and Falsifier such that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s_0,i}(\varphi)$ if and only if $\mathcal{M}, s_0, i \models \varphi$. The construction is done inductively on the structure of φ. For each state $s \in S$ and state formula φ we define a characteristic OCG $\mathcal{G}_{\mathcal{M},s}(\varphi)$.

For formulas of the form $p, \neg\varphi_1$ and $\varphi_1 \vee \varphi_2$ the construction is as for ATL assuming in the inductive cases that $\mathcal{G}_{\mathcal{M},s}(\varphi_1)$ and $\mathcal{G}_{\mathcal{M},s}(\varphi_2)$ have already been defined. The interesting case is $\varphi = \langle\langle A\rangle\rangle\varphi_1$. Here, let $\psi_1, ..., \psi_m$ be the outermost proper state subformulas of φ_1. Let $P = \{p_1, ..., p_m\}$ be fresh propositions and let $f(\varphi_1) = \varphi_1[\psi_1 \mapsto p_1, ..., \psi_m \mapsto p_m]$ be the formula obtained from φ_1 by replacing the outermost proper state subformulas with the corresponding fresh propositions. Let $\mathrm{AP}' = \mathrm{AP} \cup P$. Now, $f(\varphi_1)$ is an LTL formula over AP'. We can therefore construct a DPA $\mathcal{A}_{f(\varphi_1)}$ with input alphabet $2^{\mathrm{AP}'}$ such that the language $L(\mathcal{A}_{f(\varphi_1)})$ of the automaton is exactly the set of linear models of $f(\varphi_1)$. The number of states of the DPA can be bounded by $2^{2^{O(|f(\varphi_1)|)}}$ and the number of colors of the DPA can be bounded by $2^{O(|f(\varphi_1)|)}$ [18].

The game $\mathcal{G}_{\mathcal{M},s}(\varphi)$ is now constructed with the same structure as $\mathcal{M}$, where Verifier controls the states for players in A and Falsifier controls the states for players in $\Pi \setminus A$. However, we need to deal with truth values of the formulas $\psi_1, ..., \psi_m$ which can in general not be labelled to states in $\mathcal{M}$ since they depend both on the current state and counter value. Therefore we change the structure to obtain $\mathcal{G}_{\mathcal{M},s}(\varphi)$: For each transition $(s, v, t) \in R$ we embed a module as shown in Figure 4. Here, $2^{\mathrm{AP}'} = \{\Phi_0, ..., \Phi_\ell\}$ and for each $0 \le j \le \ell$ we let $\{\psi_{j0}, ..., \psi_{jk_j}\} = \{\psi_i \mid p_i \in \Phi_j\} \cup \{\neg\psi_i \mid p_i \notin \Phi_j\}$. Such a module is added before the initial state as well.

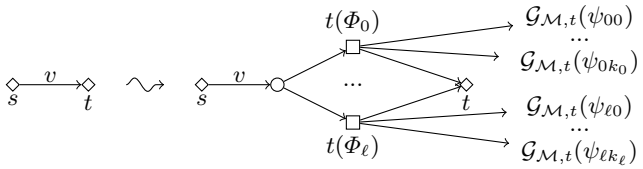

Fig. 4. $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle\varphi)$ is obtained by updating each transition as shown in the figure.

The idea is that when a transition is taken from (s, w) to $(t, w + v)$, Verifier must specify which of the propositions $p_1, ..., p_m$ are true in $(t, w + v)$, this is done by picking one of the subsets Φ_j (which is the set of propositions that are true in state $t(\Phi_j)$). Then, to make sure that Verifier does not cheat, Falsifier

has the opportunity to challenge any of the truth values of the propositions specified by Verifier. If Falsifier challenges, the play never returns again. Thus, if Falsifier challenges incorrectly, Verifier can make sure to win the game. However, if Falsifier challenges correctly then Falsifier can be sure to win the game. If Verifier has a winning strategy, then it consists in choosing the correct values of the propositions at each step. If Verifier does choose correctly and Falsifier never challenges, the winner of the game should be determined based on whether the LTL property specified by $f(\varphi_1)$ is satisfied during the play. We handle this by labelling $t(\Phi_j)$ with the propositions in Φ_j. Further, since every step of the game is divided into three steps (the original step, the specification by Verifier and the challenge opportunity for Falsifier) we alter $\mathcal{A}_{f(\varphi_1)}$ such that it only takes a transition every third step. This simply increases its size by a factor 3. We then perform a product of the game with the updated parity automaton to obtain the parity game $\mathcal{G}_{\mathcal{M},s}(\langle\langle A\rangle\rangle\varphi_1)$. It is important to note that the product with the automaton is not performed on the challenge modules (which are already colored), but only with states in the main module. This keeps the size of the game doubly-exponential in the size of the formula. It is possible to prove the following by induction on the structure of the formula. All cases except for $\langle\langle A\rangle\rangle\varphi$ are as for ATL. In the last case the proof follows the intuition outlined above.

Proposition 3. *For every OCG $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and $\varphi \in$ ATL**

$$\mathcal{M}, s, i \models \varphi \text{ if and only if Verifier has a winning strategy in } \mathcal{G}_{\mathcal{M},s,i}(\varphi)$$

The size of the model-checking game is doubly-exponential in the size of the formula for both OCGs and SOCGs. Indeed, we extend the technique to SOCGs as in the case of ATL. However, with respect to complexity, the blowup caused by the binary representation of edge weights only matters when the formula is fixed since the game is already doubly-exponential when the input formula is a parameter. Using Proposition 1 we get upper bounds on complexity of model-checking ATL*. For data complexity the lower bounds follow from data complexity of CTL. 2ExpSpace-hardness for combined complexity is proved in Section 4.

Theorem 2. *The data complexity for* ATL* *model-checking is* PSpace*-complete and* ExpSpace*-complete for OCGs and SOCGs respectively. The combined complexity is* 2ExpSpace*-complete for both OCGs and SOCGs.*

3.3 Adapting the construction to CTL*

While the model-checking game for ATL* works immediately for CTL*, the doubly-exponential size can improved. The reason is that when the model is not alternating, we can use non-deterministic Büchi automata (NBAs) for path subformulas instead of DPAs. To handle a formula of the form $\varphi = \mathbf{E}\varphi_1$ we do as for $\langle\langle A\rangle\rangle\varphi_1$ in the previous section except that the automaton $\mathcal{A}_{f(\varphi)}$ is now an NBA with $2^{O(|f(\varphi)|)}$ states [25]. Further, we need to handle the fact that the automaton is non-deterministic and therefore can have several legal transitions.

The game is simply adjusted by letting Verifier choose the transitions in the original system as well as of the automaton in each step of the main module in $\mathcal{G}_{\mathcal{M},s}(\varphi)$. This works as he just needs to show that there exists a path in the OCP along with an accepting run of the automaton in order to be sure to win. If one such exists he can show it by playing this path as well as playing the corresponding run of the automaton. The only power that Falsifier has in the main module is the possibility to challenge the values for subformulas proposed by Verifier. Thus, if Verifier proposes an incorrect valuation or plays a path that is not accepting then Falsifier can make sure to win.

Note that this construction makes the model-checking game exponential in the size of the formula. Again, Proposition 1 provides us with upper bounds. A matching ExpSpace lower bounds for combined complexity of model-checking CTL* in OCPs is shown in Section 4.

Theorem 3. *The combined complexity of model-checking* CTL* *is* ExpSpace*-complete for both OCPs and SOCPs.*

The PSpace-completeness and ExpSpace-completeness of data complexity of CTL* model-checking in OCPs and SOCPs follow immediately from results in the literature. Indeed, lower bounds are inherited from CTL model-checking results [12, 13] and upper bounds can be derived from μ-calculus results [22] as for every CTL* formula there is an equivalent μ-calculus formula. However, note that in these cases our construction above provides the matching upper bounds as well without resorting to a translation from CTL* formulas to μ-calculus formulas.

3.4 Adding counter constraints

The model-checking game can be extended to handle counter constraints by creating characteristic games $\mathcal{G}_{\mathcal{M},s}(r \bowtie c)$ for atomic formulas of the form $r \bowtie c$ with $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ where $k \in \mathbb{N}$ and $c \in \mathbb{Z}$. As examples, see $\mathcal{G}_{\mathcal{M},s}(r \leq c)$ and $\mathcal{G}_{\mathcal{M},s}(r \equiv_k c)$ illustrated in Figure 5. Using similar constructions we can handle the other cases as well. Note that adding these constraints to the logics does not increase the complexity of the algorithms in any of the cases considered.

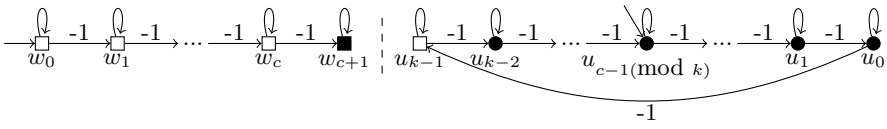

Fig. 5. $\mathcal{G}_{\mathcal{M},s}(r \leq c)$ to the left and $\mathcal{G}_{\mathcal{M},s}(r \equiv_k c)$ to the right.

Having added counter constraints it is quite easy to see that model-checking of CTL* in one-counter processes with zero-tests can be done without increasing

the complexity. This can be accomplished by updating the CTL* formula to only consider paths that are legal according to the zero-tests. By reusing a technique from [3] we can also handle systems where the counter value is allowed to be negative. Similar constructions can be made for ATL and ATL* model-checking by using alternation between players to check that no player can choose a transition that he is not allowed to choose without losing.

4 Lower bounds

For the combined complexity of CTL* in OCPs an ExpSpace lower bound does not follow immediately since the combined complexity of CTL is PSpace-complete in OCPs. To show this lower bound we do a reduction from the data complexity of CTL in SOCPs which requires a bit more work.

Proposition 4. *The combined complexity of model-checking* CTL* *in OCPs is* ExpSpace*-hard.*

Proof. We do the proof by a reduction from the model-checking problem for a fixed CTL formula in an SOCP. That is, given a CTL formula φ, an SOCP $\mathcal{M} = (S, R, \mathrm{AP}, L)$, an initial state s_0 and value $v \in \mathbb{N}$ we want to construct a CTL* formula φ' and an OCP $\mathcal{M}' = (S', R', \mathrm{AP}', L')$ such that

$$\mathcal{M}, s_0, 0 \models \varphi \text{ if and only if } \mathcal{M}', s_0, 0 \models \varphi'$$

The challenge of the construction is that $\mathcal{M}'$ can only have transitions with weights in $\{-1, 0, 1\}$. In order to accomplish this without blowing up the state space exponentially we add a module for each transition $(s, v, s') \in R$ designed to simulate adding v to the counter value. We explain the construction for $v \geq 0$ first. Let $c \in \mathbb{N}$ be the smallest number such that $2^c > w$ for every integer w that is the label of a transition in $\mathcal{M}$. Then every edge weight can be represented using c bits. Now, to obtain $\mathcal{M}'$ we do as follows. For every transition $t = (s, v, s') \in R$ with $v \geq 0$ we replace t with a module $\mathcal{M}'_t$ as shown in Figure 6.

In this module it is possible to increase the counter by any non-negative value before completing the transition from s to s'. It is even possible to stay in the module between s and s' forever (unlike in $\mathcal{M}$). Note also that v does not appear in the module at all. We will use the CTL* formula to focus on paths that behave as the transition (s, v, s') in $\mathcal{M}$ when passing through this module. A similar module can be created for $v < 0$ where the $+1$ transitions are changed to -1 transitions. We suppose that all new states in $\mathcal{M}'$ are labelled with the proposition *up* and all states from $\mathcal{M}$ are not. Further, for each state s in $\mathcal{M}'$ there is a special proposition s which is true exactly in that state.

Observe that the resulting structure $\mathcal{M}'$ is an OCP since there are only transition weights in $\{-1, 0, 1\}$ and further, the reduction is polynomial in the number of bits used to represent the integer weights in $\mathcal{M}$. We next propose a function f mapping CTL formulas to CTL* formulas such that $\mathcal{M}, s_0, 0 \models \varphi$ if and only if $\mathcal{M}', s_0, 0 \models f(\varphi)$ for every CTL formula φ. First, let

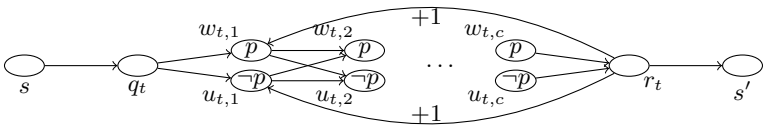

Fig. 6. Module $\mathcal{M}'_t$ for transition $t = (s, v, s')$ with $v \geq 0$.

$$\psi_{count} = \bigwedge_{t=(s,v,s')\in R} \mathbf{G}(\psi_1(t) \wedge \psi_2(t) \wedge \psi_3(t) \wedge \psi_4(t))$$

The intuition is that the path formula ψ_{count} is true along a path in $\mathcal{M}'$ if every subpath through a module $\mathcal{M}'_t$ with $t = (s, v, s')$ updates the counter by adding exactly v before reaching s'. Thus, the formula is true along a path ρ' in $\mathcal{M}'$ if and only if ρ' corresponds to a path in $\mathcal{M}$ (where each counter update of value v takes $(|v| + 1) \cdot (c + 1) + 2$ steps in ρ'). The reason that we need to enforce counter updates in this way is to not blow up the size of the OCP $\mathcal{M}'$. This is important since edge weights are exponentially large in the input in $\mathcal{M}$.

The truth value of proposition p is used as the bit-representation of the value that the counter has already been updated with where the least significant bit occurs first. The intuitive meaning of the subformulas are as follows for each transition $t = (s, a, s')$.

- $\psi_1(t)$: When the module $\mathcal{M}'_t$ is entered, the path goes through only $\neg p$ states until r_t since the counter has initially been updated with 0.
- $\psi_2(t)$: The counter value must be updated by one every time r_t is reached except the last time before the module is left.
- $\psi_3(t)$: The path must exit the module before the counter has been updated 2^c times.
- $\psi_4(t)$: If the path exits the module, the counter must have been updated exactly $|v|$ times.

The subformulas are defined in LTL as below, where $\mathbf{X}^j$ is defined inductively by $\mathbf{X}^1 = \mathbf{X}$ and $\mathbf{X}^j = \mathbf{X}^{j-1}\mathbf{X}$ for $j > 1$.

$$\psi_1(t) = q_t \rightarrow \left(\bigwedge_{i=1}^{c} \mathbf{X}^i \neg p\right)$$

$$\psi_2(t) = \left[(q_t \vee r_t) \rightarrow \bigwedge_{i=1}^{c} \left(\bigwedge_{j=1}^{i-1} \mathbf{X}^j p \leftrightarrow \left(\mathbf{X}^{i+c+1} p \leftrightarrow \mathbf{X}^i \neg p\right)\right)\right] \mathbf{U}$$
$$\mathbf{X}(\neg r_t \mathbf{U}(r_t \wedge \mathbf{X} s'))$$

$$\psi_3(t) = r_t \rightarrow \left(\bigvee_{i=1}^{c} \mathbf{X}^i p\right) \mathbf{U} \mathbf{X} s'$$

Finally, for each transition $t = (s, v, s')$ let $v' = |v|$ and let $b_1, ..., b_c$ be the c-bit representation of v' where b_1 is the least significant bit. Let B_t be the set of indices j such that $b_j = 1$ and C_t be the set of indices j such that $b_j = 0$. Now, define

$$\psi_4(t) = (q_t \vee r_t) \wedge \mathbf{X}(\neg r_t \mathbf{U}(r_t \wedge \mathbf{X}s')) \rightarrow \bigwedge_{j \in B_t} \mathbf{X}^j p \wedge \bigwedge_{j \in C_t} \mathbf{X}^j \neg p$$

We next define f inductively on the structure of a CTL formula. Thus, for every proposition q from the labelling of $\mathcal{M}$ and all CTL formulas φ_1, φ_2

$$f(q) = q$$

$$f(\varphi_1 \vee \varphi_2) = f(\varphi_1) \vee f(\varphi_2)$$

$$f(\neg\varphi_1) = \neg f(\varphi_1)$$

$$f(\mathbf{EG}\varphi_1) = \mathbf{E}(\psi_{count} \wedge \mathbf{G}(up \vee (\neg up \wedge f(\varphi_1))))$$

$$f(\mathbf{E}\varphi_1 \mathbf{U} \varphi_2) = \mathbf{E}(\psi_{count} \wedge (up \vee (\neg up \wedge f(\varphi_1))) \mathbf{U} (\neg up \wedge f(\varphi_2)))$$

$$f(\mathbf{EX}\varphi_1) = \mathbf{E}(\psi_{count} \wedge (\mathbf{X}up \mathbf{U} \mathbf{X}(\neg up \wedge f(\varphi_1))))$$

It is now possible to show by a simple induction on the structure of the CTL formula φ that for every state $s_0 \in S$ and every $v \in \mathbb{N}$ we have $\mathcal{M}, s_0, v \models \varphi$ if and only if $\mathcal{M}', s_0, v \models f(\varphi)$. □

For combined complexity of ATL* we can show that 2ExpSpace is a tight lower bound by a reduction from the word acceptance problem of a doubly-exponential space deterministic Turing machine.

Proposition 5. *The combined complexity of model-checking* ATL* *is* 2ExpSpace*-hard in both OCPs and SOCPs.*

Proof (Sketch). In [15] the emptiness problem for alternating finite automata with a one-letter alphabet (1L-AFA) is shown to be PSpace-complete by a reduction from the word acceptance problem of a polynomial space deterministic Turing machine. Another way to interpret the emptiness problem of 1L-AFAs is the problem of determining the winner in a one-counter game with two players Verifier and Falsifier where Verifier has a reachability condition and Falsifier has a safety condition. As a first step in our proof we do a similar reduction, but do it from a doubly-exponential space deterministic Turing machine $\mathcal{T}$ and an input word w to a one-counter reachability game $\mathcal{G}$ with a number of states that is doubly-exponential in the size $|w|$ of w and polynomial in the size $|\mathcal{T}|$ of $\mathcal{T}$.

In order to shrink the size of the game, we use a similar trick as in the proof of Proposition 4 to implement a counter by using an LTL objective for Verifier rather than a reachability objective. But whereas in the proof of Proposition 4

we only needed a c-bit counter capable of counting from 0 to $2^c - 1$ we now need a $2^{|w|^k}$-bit counter capable of counting from 0 to $2^{2^{|w|^k}} - 1$ where k is a constant. In order to accomplish this we can use alternation between the players, such that Falsifier can perform tests that Verifier updates the counter appropriately during the play. The details are similar to an idea from [16, 4, 5].

In total we end up with a one-counter game $\mathcal{M}$ and an LTL objective φ both of size polynomial in $|\mathcal{T}|$ and $|w|$ such that $\mathcal{T}$ accepts w if and only if the initial state of the $\mathcal{M}$ satisfies the ATL* formula $\langle\langle\{\text{Verifier}\}\rangle\rangle\varphi$ where $\varphi \in$ LTL. Note that the intermediate step with a doubly-exponential sized game is not needed but it makes the intuition and the proof much simpler. The game $\mathcal{M}$ and the LTL objective φ can be constructed directly from $\mathcal{T}$ and w in polynomial time. □

Since our lower bound is for formulas of the form $\langle\langle\{\text{I}\}\rangle\rangle\varphi$ where φ is an LTL formula and I is a player this means that the complexity of deciding the winner in one-counter games with LTL objectives is 2ExpSpace-complete both in the succinct and non-succinct case. With a fixed formula the complexity of this problem is PSpace-complete in OCGs due to PSpace-hardness of model-checking LTL in finite-state systems [23]. For SOCGs the model-checking game from Section 3 provides a reduction from data complexity of model-checking CTL in SOCPs to deciding the winner in succinct OCPGs with a fixed number of colors. Such a parity condition can be expressed by a fixed LTL objective.

Corollary 1. *Deciding the winner in two-player OCGs and SOCGs with* LTL *objectives are both* 2ExpSpace*-complete. For a fixed* LTL *formula, these problems are* PSpace*-complete and* ExpSpace*-complete respectively.*

5 Concluding remarks

Model-checking of quantitative extensions of branching and alternating-time temporal logics in one-counter systems is a very natural approach to verification and synthesis of systems with combined qualitative and quantitative objectives. In this paper we have provided complexity results for important basic problems which can help guide the direction for further research in this area.

I want to thank Valentin Goranko and Michael Reichhardt Hansen for discussions and helpful comments.

References

1. Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.
2. Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with energy constraints. In *FORMATS*, pages 33–47, 2008.
3. Patricia Bouyer, Patrick Gardy, and Nicolas Markey. Quantitative verification of weighted kripke structures. In *ATVA*, pages 64–80, 2014.

4. Laura Bozzelli. Complexity results on branching-time pushdown model checking. *Theor. Comput. Sci.*, 379(1-2):286–297, 2007.
5. Laura Bozzelli, Aniello Murano, and Adriano Peron. Pushdown module checking. *Formal Methods in System Design*, 36(1):65–95, 2010.
6. Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.
7. Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *SR*, pages 33–41, 2013.
8. Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.
9. Stéphane Demri and Régis Gascon. The effects of bounding syntactic resources on presburger ltl. *J. Log. Comput.*, 19(6):1541–1575, 2009.
10. E. Allen Emerson. The beginning of model checking: A personal perspective. In *25 Years of Model Checking - History, Achievements, Perspectives*, pages 27–45, 2008.
11. E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.
12. Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Model checking succinct and parametric one-counter automata. In *ICALP (2)*, pages 575–586, 2010.
13. Stefan Göller and Markus Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013.
14. Markus Holzer. On emptiness and counting for alternating finite automata. In *Developments in Language Theory*, pages 88–97, 1995.
15. Petr Jancar and Zdenek Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Inf. Process. Lett.*, 104(5):164–167, 2007.
16. Orna Kupferman, P. Madhusudan, P. S. Thiagarajan, and Moshe Y. Vardi. Open systems in reactive environments: Control and synthesis. In *CONCUR*, pages 92–107, 2000.
17. Christof Löding, P. Madhusudan, and Olivier Serre. Visibly pushdown games. In *FSTTCS*, pages 408–420, 2004.
18. Nir Piterman. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.
19. Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.
20. Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.
21. Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *ICALP*, pages 652–671, 1989.
22. Olivier Serre. Parity games played on transition graphs of one-counter processes. In *FoSSaCS*, pages 337–351, 2006.
23. A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.
24. Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.
25. Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *FOCS*, pages 185–194, 1983.

A Full proof of Proposition 2

Proposition 2. *For every OCG $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and $\varphi \in$ ATL*

$$\mathcal{M}, s, i \models \varphi \text{ if and only if Verifier has a winning strategy in } \mathcal{G}_{\mathcal{M},s,i}(\varphi)$$

Proof. The proof is done for QATL formulas (and thus works for ATL as well). The proof is done by induction on the structure of φ. First, we consider the base cases.

$\varphi = p$: In this case Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(p)$ if and only if $p \in L(s)$ if and only if $\mathcal{M}, s, i \models p$.

$\varphi = (r \leq c)$: In this case the counter is initially increased to i after i steps of the game. Then, Falsifier can win exactly if he can decrease the counter $c+1$ times which is possible if and only if $c + 1 \leq i$. Thus, Verifier can win exactly if $i \leq c$. By the semantics of QATL this is exactly the case when $\mathcal{M}, s, i \models r \leq c$.

$\varphi = (r < c)$: The argument is similar to the case above but the characteristic game has one state less.

$\varphi = (r \equiv_k c)$: In this case, Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(r \equiv_k c)$ if and only if he has a winning strategy where he subtracts one from the counter every time he can. The same is the case for Falsifier. For Verifier this is a winning strategy if and only if $i \equiv_k c$ if and only if $\mathcal{M}, s, i \models (r \equiv_k c)$. The reason is that after subtracting from the counter i times, the current state will be u_{k-1} if and only if

$$k - 1 \equiv c - 1 - i \pmod{k}$$

$$\Leftrightarrow k \equiv c - i \pmod{k}$$

$$\Leftrightarrow i \equiv c \pmod{k} \Leftrightarrow i \equiv_k c$$

The cases $r > c, r \geq c$ and $r = c$ can be defined using the other subformulas and thus we do not need to handle them seperately (though, this could be done using ideas similar to the above cases). Next, we consider the inductive cases.

$\varphi = \varphi_1 \vee \varphi_2$: Clearly, if Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$ or in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_2)$ then he has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1 \vee \varphi_2)$ since he can choose which of the games to play and reuse the winning strategy. On other hand, if Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1 \vee \varphi_2)$ then he is either winning in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$ or in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_2)$ because he can reuse the strategy and be sure to win in at least one of these games. Then, by using the induction hypothesis we have that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1 \vee \varphi_2)$ if and only if he has

a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$ or in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_2)$ if and only if $\mathcal{M}, s, i \models \varphi_1$ or $\mathcal{M}, s, i \models \varphi_2$ if and only if $\mathcal{M}, s, i \models \varphi_1 \vee \varphi_2$.

$\varphi = \neg\varphi_1$: The construction essentially switches Verifier with Falsifier when creating $\mathcal{G}_{\mathcal{M},s,i}(\neg\varphi_1)$ from $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$. This means that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\neg\varphi_1)$ if and only if Falsifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$. We have that one-counter games with parity conditions are determined [22]. It follows that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\neg\varphi_1)$ if and only if Verifier does not have a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$. Using the induction hypothesis this means that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\neg\varphi_1)$ if and only if $\mathcal{M}, s, i \not\models \varphi_1$ if and only if $\mathcal{M}, s, i \models \neg\varphi_1$.

$\varphi = \langle\langle A\rangle\rangle \mathbf{X}\varphi_1$: There are two cases to consider. First, suppose $s \in S_j$ for some $j \in A$. Then Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{X}\varphi_1)$ if and only if there is a transition $(s, v, s') \in R$ with $v + i \geq 0$ such that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s',i+v}(\varphi_1)$ since parity objectives are prefix independent. Using the induction hypothesis, this is the case if and only if there is a transition $(s, v, s') \in R$ with $v + i \geq 0$ such that $\mathcal{M}, s', i + v \models \varphi_1$ which is the case if and only if $\mathcal{M}, s, i \models \langle\langle A\rangle\rangle \mathbf{X}\varphi_1$. For the case where $s \notin S_j$ for all $j \in A$ the proof is similar, but uses universal quantification over the transitions.

$\varphi = \langle\langle A\rangle\rangle \mathbf{G}\varphi_1$: The intuition of the construction is that Verifier controls the players in A and Falsifier controls the players in $\Pi \setminus A$. At each configuration $(s', v) \in S \times \mathbb{N}$ of the game Falsifier can challenge the truth value of φ_1 by going to $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ in which Falsifier has a winning strategy if and only if φ_1 is indeed false in $\mathcal{M}, s', v$ by the induction hypothesis. If Falsifier challenges at the wrong time or never challenges then Verifier can make sure to win.

More precisely, suppose Verifier has a winning strategy σ in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ then every possible play when Verifier plays according to σ either never goes into one of the modules $\mathcal{G}_{\mathcal{M},s'}(\varphi_1)$ or the play goes into one of the modules at some point and never returns. Since σ is a winning strategy for Verifier, we have by the induction hypothesis that every pair $(s', v) \in S \times \mathbb{N}$ reachable when Verifier plays according to σ is such that $\mathcal{M}, s', v \models \varphi_1$, because otherwise σ would not be a winning strategy for Verifier. If coalition A follows the same strategy σ adapted to $\mathcal{M}$ then the same state, value pairs are reachable. Since for all these reachable pairs (s', v) we have $\mathcal{M}, s', v \models \varphi_1$ this strategy is a witness that $\mathcal{M}, s, i \models \langle\langle A\rangle\rangle \mathbf{G}\varphi_1$.

On the other hand, suppose that coalition A can ensure $\mathbf{G}\varphi_1$ from (s, i) using strategy σ. Then in every reachable configuration (s', v) we have $\mathcal{M}, s', v \models \varphi_1$. From this we can generate a winning strategy for Verifier in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$ that plays in the same way until (if ever) Falsifier challenges and takes a transition to a module $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ for some (s', v). Since the same configurations can be reached before a challenge as when A plays according to σ, this means that Verifier can make sure to win in $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ by the induction hypothesis. Thus, if Falsifier challenges Verifier can make sure to win and if Falsifier never

challenges Verifier also wins since all states reached have color 0. Thus, Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \mathbf{G}\varphi_1)$.

$\varphi = \langle\langle A\rangle\rangle \varphi_1 \mathbf{U} \varphi_2$: The proof works as in the case above with some minor differences. In this case, Verifier needs to show that he can reach a configuration where φ_2 is true when controlling the players in A and therefore he loses if Falsifier does not challenge incorrectly and he never reaches a module $\mathcal{G}_{\mathcal{M},s',v}(\varphi_2)$ such that $\mathcal{M}, s', v \models \varphi_2$. At the same time, he has to make sure that configurations (s', v) where $\mathcal{M}, s', v \not\models \varphi_1$ are not reached in an intermediate configuration since Falsifier still has the ability to challenge, as in the previous case. Note that Verifier gets the chance to commit to showing that φ_2 is true in a given configuration before Falsifier gets the change to challenge the value of φ_1. This is due to the definition of the until operator that does not require φ_1 to be true at the point where φ_2 becomes true. We leave out the remaining details. □

B Remaining part of proof of Proposition 3

Proposition 3. *For every OCG $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and $\varphi \in$ ATL**

$$\mathcal{M}, s, i \models \varphi \text{ if and only if Verifier has a winning strategy in } \mathcal{G}_{\mathcal{M},s,i}(\varphi)$$

Proof. We use the notation, construction and intuition from the main part of the paper. The proof is done by induction on the structure of φ. The base cases as well as boolean combinations are omitted since they work as for ATL. The interesting case is $\varphi = \langle\langle A\rangle\rangle \varphi_1$.

Suppose first that $\mathcal{M}, s, i \models \langle\langle A\rangle\rangle \varphi_1$. Then coalition A has a winning strategy σ in $\mathcal{M}$. From this, we generate a strategy σ' for Verifier in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \varphi_1)$ that consists in never cheating when specifying values of atomic formulas and choosing transitions according to what σ would have done in $\mathcal{M}$. Then, if Falsifier challenges at some point, Verifier can be sure to win by the induction hypothesis since he never cheats. If Falsifier never challenges (or, until he challenges), Verifier simply mimics the collective winning strategy σ of coalition A in $\mathcal{M}$ from (s, i). This ensures that he wins in the parity game due to the definition of the parity condition from the parity automaton corresponding to $f(\varphi_1)$.

Suppose on the other hand that in $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \varphi_1)$ Verifier has a winning strategy σ. Then σ never cheats when specifying values of propositions, because then Falsifier could win according to the induction hypothesis. Define a strategy σ' for coalition A in $\mathcal{M}$ that plays like σ in the part of $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \varphi_1)$ where no challenge has occured. σ' is winning for A with condition φ_1 in $\mathcal{M}$ due to the definition of $\mathcal{G}_{\mathcal{M},s,i}(\langle\langle A\rangle\rangle \varphi_1)$ using the automaton $\mathcal{A}_{f(\varphi_1)}$. □

C Full proof of Proposition 4

Proposition 4. *The combined complexity of model-checking* CTL* *in OCPs is* ExpSpace*-hard.*

Proof. We do the proof by a reduction from the model-checking problem for a fixed CTL formula in an SOCP. That is, given a CTL formula φ, an SOCP $\mathcal{M} = (S, R, \mathrm{AP}, L)$, an initial state s_0 and value $v \in \mathbb{N}$ we want to construct a CTL* formula φ' and an OCP $\mathcal{M}' = (S', R', \mathrm{AP}', L')$ such that

$$\mathcal{M}, s_0, 0 \models \varphi \text{ if and only if } \mathcal{M}', s_0, 0 \models \varphi'$$

The challenge of the construction is that $\mathcal{M}'$ can only have transitions with weights in $\{-1, 0, 1\}$. In order to accomplish this without blowing up the state space exponentially we add a module for each transition $(s, v, s') \in R$ designed to simulate adding v to the counter value. We explain the construction for $v \geq 0$ first. Let $c \in \mathbb{N}$ be the smallest number such that $2^c > w$ for every integer w that is the label of a transition in $\mathcal{M}$. Then every edge weight can be represented using c bits. Now, to obtain $\mathcal{M}'$ we do as follows. For every transition $t = (s, v, s') \in R$ with $v \geq 0$ we replace t with a module $\mathcal{M}'_t$ as shown in Figure 7. In this module it is possible to increase the counter by any non-negative value before completing the transition from s to s'. It is even possible to stay in the module between s and s' forever (unlike in $\mathcal{M}$). Note also that v does not appear in the module at all. We will use the CTL* formula to focus on paths that behave as the transition (s, v, s') in $\mathcal{M}$ when passing through this module. A similar module can be created for $v < 0$ where the $+1$ transitions are changed to -1 transitions. We suppose that all new states in $\mathcal{M}'$ are labelled with the proposition *up* and all states from $\mathcal{M}$ are not. Further, for each state s in $\mathcal{M}'$ there is a special proposition s which is true exactly in that state.

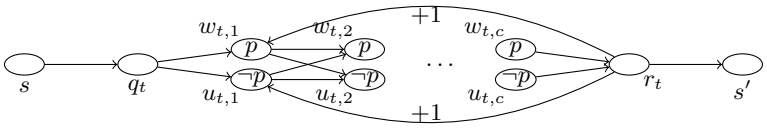

Fig. 7. Module $\mathcal{M}'_t$ for transition $t = (s, v, s')$ with $v \geq 0$.

Observe that the resulting structure $\mathcal{M}'$ is an OCP since there are only transition weights in $\{-1, 0, 1\}$ and further, the reduction is polynomial in the number of bits used to represent the integer weights in $\mathcal{M}$. We next propose a function f mapping CTL formulas to CTL* formulas such that $\mathcal{M}, s_0, 0 \models \varphi$ if and only if $\mathcal{M}', s_0, 0 \models f(\varphi)$ for every CTL formula φ. First, let

$$\psi_{count} = \bigwedge_{t=(s,v,s') \in R} \mathbf{G}(\psi_1(t) \wedge \psi_2(t) \wedge \psi_3(t) \wedge \psi_4(t))$$

The intuition is that the path formula ψ_{count} is true along a path in $\mathcal{M}'$ if every subpath through a module $\mathcal{M}'_t$ with $t = (s, v, s')$ updates the counter by adding exactly v before reaching s'. Thus, the formula is true along a path ρ'

in $\mathcal{M}'$ if and only if ρ' corresponds to a path in $\mathcal{M}$ (where each counter update of value v takes $(|v|+1)\cdot(c+1)+2$ steps in ρ'). The reason that we need to enforce counter updates in this way is to not blow up the size of the OCP $\mathcal{M}'$. This is important since edge weights are exponentially large in the input in $\mathcal{M}$.

The truth value of proposition p is used as the bit-representation of the value that the counter has already been updated with where the least significant bit occurs first. The intuitive meaning of the subformulas are as follows for each transition $t = (s, a, s')$.

- $\psi_1(t)$: When the module $\mathcal{M}'_t$ is entered, the path goes through only $\neg p$ states until r_t since the counter has initially been updated with 0.
- $\psi_2(t)$: The counter value must be updated by one every time r_t is reached except the last time before the module is left.
- $\psi_3(t)$: The path must exit the module before the counter has been updated 2^c times.
- $\psi_4(t)$: If the path exits the module, the counter must have been updated exactly $|v|$ times.

The subformulas are defined in LTL as below, where $\mathbf{X}^j$ is defined inductively by $\mathbf{X}^1 = \mathbf{X}$ and $\mathbf{X}^j = \mathbf{X}^{j-1}\mathbf{X}$ for $j > 1$.

$$\psi_1(t) = q_t \rightarrow \left(\bigwedge_{i=1}^{c} \mathbf{X}^i \neg p\right)$$

$$\psi_2(t) = \left[(q_t \vee r_t) \rightarrow \bigwedge_{i=1}^{c}\left(\bigwedge_{j=1}^{i-1} \mathbf{X}^j p \leftrightarrow \left(\mathbf{X}^{i+c+1} p \leftrightarrow \mathbf{X}^i \neg p\right)\right)\right] \mathbf{U}$$
$$\mathbf{X}(\neg r_t \mathbf{U}(r_t \wedge \mathbf{X} s'))$$

$$\psi_3(t) = r_t \rightarrow \left(\bigvee_{i=1}^{c} \mathbf{X}^i p\right) \mathbf{U}\mathbf{X} s'$$

Finally, for each transition $t = (s, v, s')$ let $v' = |v|$ and let $b_1, ..., b_c$ be the c-bit representation of v' where b_1 is the least significant bit. Let B_t be the set of indices j such that $b_j = 1$ and C_t be the set of indices j such that $b_j = 0$. Now, define

$$\psi_4(t) = (q_t \vee r_t) \wedge \mathbf{X}(\neg r_t \mathbf{U}(r_t \wedge \mathbf{X} s')) \rightarrow \bigwedge_{j \in B_t} \mathbf{X}^j p \wedge \bigwedge_{j \in C_t} \mathbf{X}^j \neg p$$

We now define f inductively on the structure of a CTL formula. Thus, for every proposition q from the labelling of $\mathcal{M}$ and all CTL formulas φ_1, φ_2

$$f(q) = q$$

$$f(\varphi_1 \vee \varphi_2) = f(\varphi_1) \vee f(\varphi_2)$$

$f(\neg\varphi_1) = \neg f(\varphi_1)$

$f(\mathbf{EG}\varphi_1) = \mathbf{E}(\psi_{count} \wedge \mathbf{G}(up \vee (\neg up \wedge f(\varphi_1))))$

$f(\mathbf{E}\varphi_1\mathbf{U}\varphi_2) = \mathbf{E}(\psi_{count} \wedge (up \vee (\neg up \wedge f(\varphi_1)))\mathbf{U}(\neg up \wedge f(\varphi_2)))$

$f(\mathbf{EX}\varphi_1) = \mathbf{E}(\psi_{count} \wedge (\mathbf{X}up\mathbf{U}\mathbf{X}(\neg up \wedge f(\varphi_1))))$

Now, by induction on the structure of the CTL formula φ we show that for every state $s_0 \in S$ and every $v \in \mathbb{N}$ we have $\mathcal{M}, s_0, v \models \varphi$ if and only if $\mathcal{M}', s_0, v \models f(\varphi)$.

$\varphi = q$: For the base case it is true immediately since for every state $s \in S$ and $q \in \mathrm{AP}$ we have $q \in L(s)$ iff $q \in L'(s)$.

Assume as induction hypothesis that the claim is true for all proper subformulas of φ. Then we have the following cases.

$\varphi = \varphi_1 \vee \varphi_2$: By the induction hypothesis we have

$\mathcal{M}, s_0, v \models \varphi_1 \vee \varphi_2$

iff $\mathcal{M}, s_0, v \models \varphi_1$ or $\mathcal{M}, s_0, v \models \varphi_2$

iff $\mathcal{M}', s_0, v \models f(\varphi_1)$ or $\mathcal{M}', s_0, v \models f(\varphi_2)$

iff $\mathcal{M}', s_0, v \models f(\varphi_1) \vee f(\varphi_2)$

$\varphi = \neg\varphi_1$: By the induction hypothesis we have

$\mathcal{M}, s_0, v \models \neg\varphi_1$

iff $\mathcal{M}, s_0, v \not\models \varphi_1$

iff $\mathcal{M}', s_0, v \not\models f(\varphi_1)$

iff $\mathcal{M}', s_0, v \models \neg f(\varphi_1)$

iff $\mathcal{M}', s_0, v \models f(\neg\varphi_1)$

$\varphi = \mathbf{EG}\varphi_1$: Suppose first that $\mathcal{M}, s_0, v \models \mathbf{EG}\varphi_1$. Then there exists a path ρ in $\mathcal{M}$ from s_0 such that φ_1 is true in every configuration of ρ. There exists a corresponding path in $\mathcal{M}'$ which passes through the same configurations as ρ but with update modules in between. Let ρ' be the unique path with this property and such that ρ' also satisfies ψ_{count}. Now, in every configuration of

ρ' either up is true or false. When up is false the current configuration of ρ' is also a configuration of ρ since ρ' satisfies ψ_{count}. By the induction hypothesis, each such configuration of ρ' thus satisfies $f(\varphi_1)$. This means that we have $\mathcal{M}', s_0, v \models \mathbf{E}(\psi_{count} \wedge \mathbf{G}(up \vee (\neg up \wedge f(\varphi_1)))) = f(\mathbf{EG}\varphi_1)$. For the other direction the argument is similar, showing that if there is a path ρ' in $\mathcal{M}'$ from s_0, v satisfying $\psi_{count} \wedge \mathbf{G}(up \vee (\neg up \wedge f(\varphi_1)))$ then there exists a corresponding path ρ in $\mathcal{M}$ from s_0, v satisfying $\mathbf{EG}\varphi_1$.

The proof for $\mathbf{E}(\varphi_1 \mathbf{U} \varphi_2)$ and $\mathbf{EX}\varphi_1$ are done similarly to the case above. Thus, we have that $\mathcal{M}, s_0, v \models \varphi$ iff $\mathcal{M}', s_0, v \models f(\varphi)$. In particular this is the case for $v = 0$. Since the model-checking problem for CTL in SOCPs can be easily reduced to the same problem where the initial value $v = 0$ then we have that the model-checking problem for CTL* in OCPs is ExpSpace-hard since the constructions of $\mathcal{M}'$ and $f(\varphi)$ above are polynomial. Note that the CTL* formula is not fixed even if the CTL formula is and therefore the result does not apply for the data complexity of CTL*. □

D Full proof of Proposition 5

We will show that model-checking ATL* in OCGs is 2ExpSpace-hard by a reduction from the word acceptance problem for a deterministic doubly-exponential space Turing machine. From this, the theorem follows from the observations in the main text.

Let $\mathcal{T} = (Q, q_0, \Sigma, \delta, q_F)$ be a deterministic Turing machine that uses at most $2^{2^{|w|^k}}$ tape cells on input w where k is a constant and $|w|$ is the number of symbols in w. Here, Q is a finite set of control states, $q_0 \in Q$ is the initial control state. $\Sigma = \{0, 1, \#, a, r\}$ is the tape alphabet containing the blank symbol $\#$ and special symbols a and r such that $\mathcal{T}$ accepts immediately if it reads a and rejects immediately if it reads r, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{\text{Left}, \text{Right}\}$ is the transition function and $q_F \in Q$ is the accepting state. If $\delta(q, a) = (q', a', x)$ we write $\delta_1(q, a) = q', \delta_2(q, a) = a'$ and $\delta_3(q, a) = x$. Let $\Sigma_I = \Sigma \setminus \{\#\}$. Now, let $w = w_1 ... w_{|w|} \in \Sigma_I^*$ be an input word. From this we construct an OCG $\mathcal{M}$, an initial state s_0 and an ATL* formula Φ all with size polynomial in $n = |w|^k$ and $|\mathcal{T}|$ such that $\mathcal{T}$ accepts w if and only if $\mathcal{M}, (s_0, 0) \models \Phi$.

We use an intermediate step in the reduction for simplicity. This is done by considering an OCG $\mathcal{G} = (S', \{\text{Verifier}, \text{Falsifier}\}, (S'_{\text{Verifier}}, S'_{\text{Falsifier}}), R')$ with two players Verifier and Falsifier and an initial state s'_0 such that Verifier can force the play to reach s'_F if and only if $\mathcal{T}$ accepts w. However, the size of the set S' of states will be doubly-exponential in n. The idea of this construction resembles a reduction from the word acceptance problem for polynomial-space Turing machines to the emptiness problem for alternating finite automata with a singleton alphabet used in [15]. Afterwards we will reduce this to model-checking of the ATL* formula Φ in $\mathcal{M}$ where $|S|$ is polynomial in n. This reduction can be performed by considering a more involved formula. We will use a technique similar to [16, 4, 5] to simulate a 2^n-bit counter by using LTL properties and

alternation between the players. This is the main trick to keep the state-space of $\mathcal{M}$ small.

We start with some notation. We assume that $\mathcal{T}$ uses the tape cells numbered $1, ..., 2^{2^n}$ and that the tape head points to position 1 initially. In addition, suppose for ease of arguments that there are two extra tape cells numbered 0 and $2^{2^n}+1$ such that $\mathcal{T}$ immediately accepts if the tape head reaches cell 0 or cell $2^{2^n}+1$. That is, cell 0 and $2^{2^n}+1$ holds the symbol a initially. Further, assume without loss of generality that if $\mathcal{T}$ halts it always does so with the tape head pointing to cell 1 that contains the symbol a. Since $\mathcal{T}$ is deterministic it has a unique (finite or infinite) run on the word w which is a sequence $C_0^w C_1^w ...$ of configurations. Let $\Delta = \Sigma \cup (Q \times \Sigma)$. Then each configuration C_i^w is a sequence in $\Delta^{2^{2^n}+2}$ containing exactly one element in $Q \times \Sigma$ which is used to specify the current control state and location of the tape head. For instance, the initial configuration C_0^w is given by

$$C_0^w = a(q_0, w_1)w_2w_3...w_{|w|}\#\#....\#a$$

We use $C_i^w(j)$ to denote the jth element of configuration C_i^w. For a given element $d \in \Delta$ we define the set $\mathrm{Pre}(d)$ of predecessor triples of d as

$$\begin{array}{l}
\mathrm{Pre}(d) = \{(d_1, d_2, d_3) \in \Sigma^3 \mid d_2 = d\} \\
\cup \{((q, b), d_2, d_3) \in (Q \times \Sigma) \times \Sigma^2 \mid d = (\delta_1(q, b), d_2) \text{ and } \delta_3(q, b) = \text{Right}\} \\
\cup \{((q, b), d_2, d_3) \in (Q \times \Sigma) \times \Sigma^2 \mid d = d_2 \text{ and } \delta_3(q, b) \neq \text{Right}\} \\
\cup \{(d_1, d_2, (q, b)) \in \Sigma^2 \times (Q \times \Sigma) \mid d = (\delta_1(q, b), d_2) \text{ and } \delta_3(q, b) = \text{Left}\} \\
\cup \{(d_1, d_2, (q, b)) \in \Sigma^2 \times (Q \times \Sigma) \mid d = d_2 \text{ and } \delta_3(q, b) \neq \text{Left}\} \\
\cup \{(d_1, (q, b), d_3) \in \Sigma \times (Q \times \Sigma) \times \Sigma \mid d = \delta_2(q, b)\}
\end{array}$$

The idea is that given the three elements $C_i^w(j-1), C_i^w(j)$ and $C_i^w(j+1)$ one can uniquely determine $C_{i+1}^w(j)$ according to the definition of a Turing machine. $\mathrm{Pre}(d)$ is then the set of all triples (d_1, d_2, d_3) such that it is possible to have $C_i^w(j-1) = d_1, C_i^w(j) = d_2, C_i^w(j+1) = d_3$ and $C_{i+1}^w(j) = d$.

We now define $\mathcal{G} = ((S', \{\text{Verifier}, \text{Falsifier}\}, (S'_{\text{Verifier}}, S'_{\text{Falsifier}}), R'))$ by

- $S' = (\{0, ..., 2^{2^n}+1\} \times (\Delta \cup \Delta^3)) \cup \{s'_0, s'_z, s'_r, s'_F\}$
- $S'_{\text{Verifier}} = (\{0, ..., 2^{2^n}+1\} \times \Delta) \cup \{s'_0\}$
- $S'_{\text{Falsifier}} = (\{0, ..., 2^{2^n}+1\} \times \Delta^3) \cup \{s'_z, s'_r, s'_F\}$
- R' is the least relation such that
  - $(s'_0, 1, s'_0) \in R'$
  - $(s'_0, 0, (1, (q_F, a))) \in R'$
  - $((j, d), 0, (j, (d_1, d_2, d_3))) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $(d_1, d_2, d_3) \in \mathrm{Pre}(d)$
  - For $j \in \{0, 2^{2^n}+1\}$ we have $((j, a), 0, s'_F) \in R'$ and $((j, d), 0, s'_r) \in R'$ when $d \neq a$
  - $((j, d), 0, s'_z) \in R'$ for all (j, d) such that $C_0^w(j) = d$.
  - $(s'_z, 0, s'_F) \in R'$
  - $(s'_z, -1, s'_r) \in R'$

- $((j,(d_1,d_2,d_3)),-1,(j-1,d_1)) \in R'$ for all $j \in \{1,...,2^{2^n}\}$ and all $d_1,d_2,d_3 \in \Delta$
- $((j,(d_1,d_2,d_3)),-1,(j,d_2)) \in R'$ for all $j \in \{1,...,2^{2^n}\}$ and all $d_1,d_2,d_3 \in \Delta$
- $((j,(d_1,d_2,d_3)),-1,(j+1,d_3)) \in R'$ for all $j \in \{1,...,2^{2^n}\}$ and all $d_1,d_2,d_3 \in \Delta$

The different types of transitions are shown in Figure 8, 9 and 10. The intuition is that Verifier tries to show that $\mathcal{T}$ accepts w and Falsifier tries to prevent this. Initially, Verifier can increase the counter to any natural number, assume he chooses v. If $\mathcal{T}$ accepts w it does so in a final configuration with the tape head pointing at cell 1 holding the symbol a with the current control state q_F. The game is now played by moving backwards from the state $(1,(q_F,a))$ holding this information. Verifier can choose a predecessor triple that leads to $(1,(q_F,a))$. Player Falsifier then chooses one of the elements of the triple, the counter is decreased by one and the play continues like this. Finally, if the counter is 0 in a state (j,d) such that $C_0^w(j)=d$ then Verifier can win by going to s'_z from which Falsifier can only go to s'_F. We will argue that Verifier can make sure that this happens if and only if $\mathcal{T}$ accepts w after performing v steps.

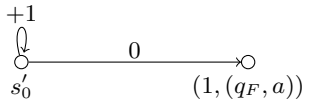

Fig. 8. From the initial state, Verifier can increase the counter to any natural number before starting the game.

Lemma 1. *The configuration* $((j,d),i) \in (\{1,...,2^{2^n}\} \times \Delta) \times \mathbb{N}$ *is winning for* Verifier *if and only if* $C_i^w(j)=d$. *In particular* $((1,(q_F,a)),i)$ *is winning for* Verifier *if and only if* $C_i^w(1)=(q_F,a)$ *if and only if* $\mathcal{T}$ *accepts* w *after* i *steps of computation.*

Proof. The proof is done by induction on i. For the base case $i=0$ the statement says that $((j,d),0)$ is winning for Verifier if and only if $C_0^w(j)=d$. Indeed, if $((j,d),0)$ is winning for Verifier he must go directly from (j,d) to s'_z because all other paths are blocked after one step since the counter value is 0. If he goes to s'_z then he wins because Falsifier can only go to s'_F. However, note that there is only a transition from (j,d) to s'_z if $C_0^w(j)=d$ by construction. Thus, if Verifier is winning from $((j,d),0)$ then $C_0^w(j)=d$. For the other direction, suppose $C_0^w(j)=d$. Then Verifier can make sure to win by going to s'_z.

For the induction step, suppose the lemma is true for i. Now we need to show that $((j,d),i+1)$ is winning for Verifier if and only if $C_{i+1}^w(j)=d$. Suppose first that $((j,d),i+1)$ is winning for Verifier. The winning strategy σ cannot

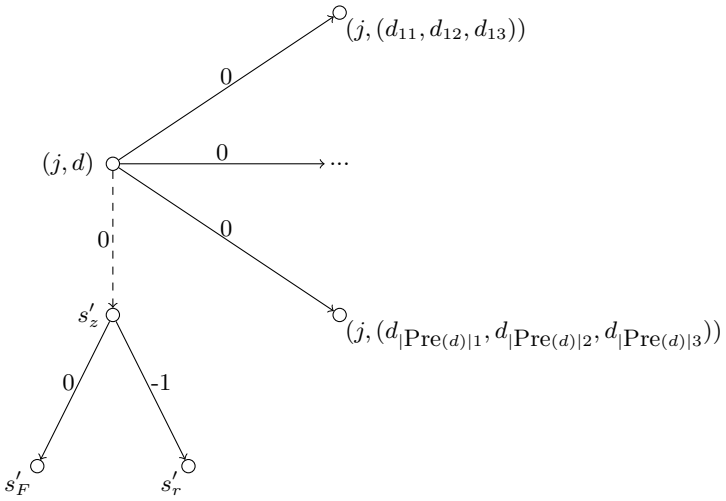

Fig. 9. From a state $(j, d) \in \{1, ..., 2^{2^n}\} \times \Delta$ Verifier can choose a predecessor triple of d. The dashed transition is enabled only when $C_0^w(j) = d$. In this case Verifier can be sure to win if the current counter value is 0.

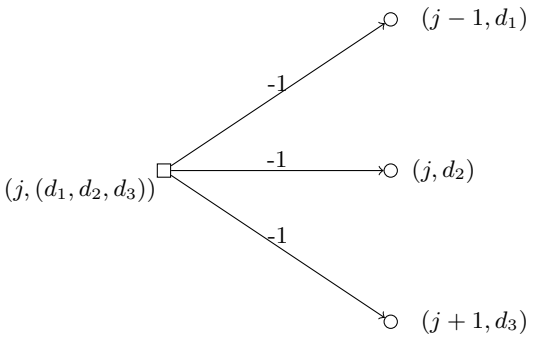

Fig. 10. From a precedessor triple chosen by Verifier, Falsifier can choose which predecessor to continue with.

consist in going directly to s'_z because then Falsifier can go to s'_r. Thus, Verifier must choose a predecessor triple $(d_1, d_2, d_3) \in \mathrm{Pre}(d)$ when playing according to σ. After he chooses this, Falsifier chooses one of them and the counter is decreased by one. Thus, Falsifier can choose either $((j-1, d_1), i)$, $((j, d_2), i)$ or $(j+1, d_3), i)$. Thus, by the induction hypothesis $C_i^w(j-1) = d_1$, $C_i^w(j) = d_2$ and $C_i^w(j+1) = d_3$ since Verifier is winning. By the definition of predecessor triples, this means that $C_{i+1}^w(j) = d$. For the other direction, suppose $C_{i+1}^w(j) = d$. Then by going to the state $(j, (C_i^w(j-1), C_i^w(j), C_i^w(j+1)))$ he can be sure to win by the induction hypothesis. □

Lemma 2. *Starting in configuration* $(s'_0, 0)$ Verifier *can make sure to reach* s'_F *if and only if* $\mathcal{T}$ *accepts* w*.*

We have now reduced the word acceptance problem to a reachability game in an OCG $\mathcal{G}$ with a doubly-exponential number of states. Due to the structure of $\mathcal{G}$ we can reduce this to model-checking the ATL* formula Φ in the OCG $\mathcal{M}$. The difficult part is that we need to store the number of the tape cell that the tape head is pointing at, which can be of doubly-exponential size. The other features of $\mathcal{G}$ are polynomial in the input. Note that at each step of the game, the position of the tape head either stays the same, increases by one or decreases by one. This is essential for our ability to encode it using ATL*. We construct $\mathcal{M}$ much like $\mathcal{G}$ but where the position of the tape head is not present in the set of states. Instead, for each transition in the game between states s and s' we have a module in which Verifier encodes the position of the tape head by his choices. This is done much like in the proof of Proposition 4. However, we need a 2^n-bit counter in this case rather than just a c-bit counter. For this we need alternation between the players. This is done by giving Falsifier the possibility to challenge if Verifier has not chosen the correct value of the tape head position. This can be ensured by use of the ATL* formula $\Phi = \langle\langle \{\text{Verifier}\} \rangle\rangle \varphi$ where φ is an LTL formula. The details of simulating a 2^n-bit counter like this can be obtained from [16, 4, 5]. According to the choices of Falsifier then Verifier must be able to increase, decrease or leave unchanged the position of the tape head. This can be enforced by a formula with a size polynomial in n. Except for having to implement the position of the tape head in this way, the rules of $\mathcal{M}$ are the same as for $\mathcal{G}$ where Verifier needs to show that $\mathcal{T}$ accepts w by choosing a strategy that ensures reaching a certain state in the game while updating the tape head position correctly. In the end, this means that for the initial state s_0 in $\mathcal{M}$ corresponding to s'_0 in $\mathcal{G}$ we get $\mathcal{M}, s_0, 0 \models \langle\langle \{\text{Verifier}\} \rangle\rangle (\varphi \wedge \mathbf{F} s_F)$ if and only if $\mathcal{T}$ halts on w. Here we assume that the play also goes to a halting state s_F corresponding to s'_F if Falsifier challenges the counter value incorrectly.

Proposition 5. *The combined complexity of model-checking* ATL* *is* 2ExpSpace*-hard in both OCPs and SOCPs.*